B3

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 06-274524
(43)Date of publication of application : 30.09.1994

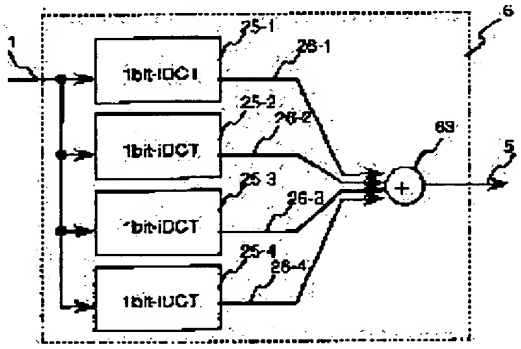
(51)Int.Cl. G06F 15/332
G06F 15/66
H03M 7/30
H04N 1/41
H04N 7/133

(21)Application number : 05-061613
(22)Date of filing : 22.03.1993

(71)Applicant : HITACHI LTD
(72)Inventor : KIMURA JUNICHI

(54) ORTHOGONAL TRANSFORMATION CIRCUIT AND INVERSE TRANSFORMATION CIRCUIT

(57)Abstract:
PURPOSE: To execute transformation or inverse transformation processings at an operation speed same as before without increasing a circuit scale by dividing data used for integrating into 1-bit data, replacing an integrating part with shift calculation and adding the data of respective bits after a product-sum operation.
CONSTITUTION: 1-bit iDCT circuits 25-1-25-4 are parallelly arranged and also an adder 63 for adding the arithmetic results 26-1-26-4 is arranged. That is, one of the 1-bit iDCT circuits 25-1-25-4 is allocated to each matrix, input signals 1 are calculated for each matrix broken down by each bit, the respective results 26-1-26-4 are added at the adder 63 and a final result is obtained. In this case, the 1-bit iDCT circuits 25-1-25-4 are circuits for integrating input vectors and the matrix for which the bit number of '1' included when the absolute values of the respective components of the matrix is binary displayed is equal to or less than 1 and are composed of shift circuits and adding-subtracting circuits.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12)**公開特許公報（Ａ）**　(11)特許出願公開番号

特開平6－274524

(43)公開日　平成６年(1994)９月30日

(51)Int.Cl.[5]		識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ	15/332		Ｓ　7343－5Ｌ		
	15/66	３３０	Ｈ　8420－5Ｌ		
Ｈ０３Ｍ	7/30		Ａ　8522－5Ｊ		
Ｈ０４Ｎ	1/41		Ｂ　9070－5Ｃ		
	7/133		Ｚ		

審査請求　未請求　請求項の数６　ＯＬ　（全 11 頁）

(21)出願番号　特願平5－61613

(22)出願日　平成５年(1993)３月22日

(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地

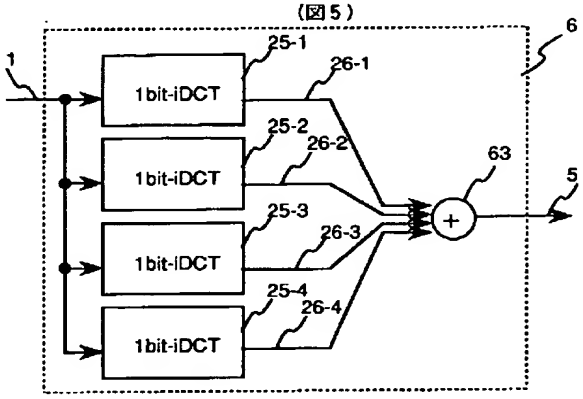
(72)発明者　木村　淳一
東京都国分寺市東恋ケ窪１丁目280番地
株式会社日立製作所中央研究所内

(74)代理人　弁理士　小川　勝男

(54)【発明の名称】　直交変換回路および逆変換回路

(57)【要約】
【構成】１ビットｉＤＣＴ回路２５－１～４を並列に配置し、それらの演算結果２６－１～４を加算する加算器６３より構成される。１ビットｉＤＣＴ回路は行列の各要素の絶対値を２進数表示したときに含まれる“１”のビット数が一つ以下の行列と入力ベクトルの積算を行う回路であり、シフト回路，加減算回路からなる。
【効果】直交変換の行列演算を１ビット毎に計算することにより従来と同じ動作速度で、しかも、回路規模も増加させることなしに変換あるいは逆変換処理を実行することができる。

(図５)



1

【特許請求の範囲】

【請求項１】Ｎ次のベクトルを入力し、入力ベクトルと直交行列の積算を行いＮ次の出力行列を得る直交変換回路において、直交行列に予め定めた係数値を掛け正規化する手段，正規化された行列の各要素の絶対値を２進数表示したときに含まれる“１”のビット数が一つ以下である行列の和あるいは差の形に分割する手段，分割された行列と入力ベクトルの積を計算する手段，各分割された行列との積算結果のベクトルを分割した手順に従って合成する手段，合成したベクトルに予め定められた出力係数を掛ける手段を具備することを特徴とする直交変換回路。

【請求項２】Ｎ次の直交変換されたベクトルを入力し、入力ベクトルと直交行列の積算を行いＮ次の出力行列を得る直交変換の逆変換回路において、直交行列に予め定めた係数値を掛け正規化する手段，正規化された行列の各要素の絶対値を２進数表示したときに含まれる“１”のビット数が一つ以下である行列の和あるいは差の形に分割する手段，分割された行列と入力ベクトルの積を計算する手段，各分割された行列との積算結果のベクトルを分割した手順に従って合成する手段，合成したベクトルに予め定められた出力係数を掛ける手段を具備することを特徴とする逆変換回路。

【請求項３】請求項１において、回路を縦続に接続し、Ｎ×Ｎ次の直交変換を行う直交変換回路。

【請求項４】請求項２において、回路を縦続に接続し、Ｎ×Ｎ次の逆変換を行う逆変換回路。

【請求項５】請求項１のベクトル合成において、分割し＊

2

＊た数よりも少ないベクトルを用いて合成を行う手段を具備する直交変換回路。

【請求項６】請求項２のベクトル合成において、分割した数よりも少ないベクトルを用いて合成を行う手段を具備する逆変換回路。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は直交変換を用いた符号化あるいは復号化処理に係り、特に、画像信号の符号化，復号化のような大量データの高速な処理を可能とする回路に関する。

【０００２】

【従来の技術】直交変換は予め定められた個数の信号を、同じ個数のそれぞれ直交するベクトルの成分に分解する手法である。直交変換の代表的なものに離散コサイン変換（以下ＤＣＴ）がある。ＤＣＴに関しての詳細は、例えばケー　アール　ラオ(K.R.Rao)他著、安田浩／藤原洋共訳「画像符号化技術－ＤＣＴとその国際標準－(原著名：Discrete Cosine Transform Algorithms, Advantages,Applications)」（オーム社、１９９２．７）全編にわたって記されている。以下、ＤＣＴの説明を、本発明に関する部分のみ簡単に行う。数１および数３は８次のＤＣＴの例である。ＤＣＴでは数１に示すようにＣＯＳで表現されるベクトル成分に分解される。ＤＣＴの逆変換（ｉＤＣＴ）は数２によって表される。

【０００３】

【数１】

$$F(u)=\frac{1}{2}C(u)\sum_{x=0}^{7} f(x)\cos\left\{\frac{(2x+1)u\times\pi}{16}\right\} \quad \cdots (数1)$$

【０００４】　※　※【数２】

$$f(x)=\frac{1}{2}\sum_{u=0}^{7} C(u)F(u)\cos\left\{\frac{(2x+1)u\times\pi}{16}\right\} \quad \cdots (数2)$$

【０００５】　★　★【数３】

$$C(u)=\begin{bmatrix} \frac{1}{\sqrt{2}} \cdots \text{for } u,v=0 \\ 1 \cdots \text{otherwise} \end{bmatrix} \quad \cdots (数3)$$

【０００６】数１のＤＣＴおよび数２のｉＤＣＴは入力ベクトルとＣＯＳで表現される８×８の行列との積によっても表現できる。数４ないし数８は数１のＤＣＴ、数２のｉＤＣＴを行列表現で表したものである。数８は数２のＣＯＳ部分、Ｃ（ｕ）および１／２の係数部分をすべてまとめたものである。数７の行列は正規直交行列であるため逆行列は元の行列の転置の形になり、ＤＣＴおよびｉＤＣＴの計算は同じ手順で計算する事ができる。

【０００７】

【数４】

$$F=A\cdot f \quad \cdots (数4)$$

【０００８】

【数５】

$$f=A^{-1}\cdot F=A^{T}\cdot F \quad \cdots (数5)$$

【０００９】

【数６】

$$f=\begin{bmatrix} f(0) \\ f(1) \\ \cdots \\ f(7) \end{bmatrix} \quad \cdots（数６）$$

$$F=\begin{bmatrix} F(0) \\ F(1) \\ \cdots \\ F(7) \end{bmatrix} \quad \cdots（数７）$$

【００１０】
【数７】

$$f=A^{T}*F*A \quad \cdots（数８）$$

$$A^{T}=\begin{bmatrix} 0.3536 & 0.4904 & 0.4619 & 0.4157 & 0.3536 & 0.2778 & 0.1913 & 0.0975 \\ 0.3536 & 0.4157 & 0.1913 & -0.0975 & -0.3536 & -0.4904 & -0.4619 & -0.2778 \\ 0.3536 & 0.2778 & -0.1913 & -0.4904 & -0.3536 & 0.0975 & 0.4619 & 0.4157 \\ 0.3536 & 0.0975 & -0.4619 & -0.2778 & 0.3536 & 0.4157 & -0.1913 & -0.4904 \\ 0.3536 & -0.0975 & -0.4619 & 0.2778 & 0.3536 & -0.4157 & -0.1913 & 0.4904 \\ 0.3536 & -0.2778 & -0.1913 & 0.4904 & -0.3536 & -0.0975 & 0.4619 & -0.4157 \\ 0.3536 & -0.4157 & 0.1913 & 0.0975 & -0.3536 & 0.4904 & -0.4619 & 0.2778 \\ 0.3536 & -0.4904 & 0.4619 & -0.4157 & 0.3536 & -0.2778 & 0.1913 & -0.0975 \end{bmatrix}$$

【００１１】
【数８】

【００１２】ＤＣＴは主に画像信号の符号化に用いられる。画像信号に用いられるＤＣＴは数１，数２に示した８次のものが多く、実際には、数８に示されるように、水平方向・垂直方向に対し変換を施す２次元のものが主に用いられる。即ち水平８画素，垂直８画素の６４画素のブロックに対を６４個のそれぞれ直交する周波数成分に分解する。逆変換（ｉＤＣＴ）は周波数成分を元の画像に変換する手法である。

【００１３】数９では６４画素の変換に４０９６回の積※

$$f(x,y)=\frac{1}{4}\sum_{u=0}^{7}\sum_{v=0}^{7}C(u)C(v)F(u,v)\cos\left\{\frac{(2x+1)u\times\pi}{16}\right\}\cos\left\{\frac{(2y+1)v\times\pi}{16}\right\} \quad \cdots（数９）$$

※和演算が必要であるが、一般には数１０のように１次元の行列演算の形に変形することにより、積和演算の数を1024回に減らすとともに数４から数８の１次元のＤＣＴ，ｉＤＣＴに帰着させることが出来る。２次元のＤＣＴ，ｉＤＣＴの高速化，簡略化は１次元のＤＣＴ，iDCTの高速化，簡略化を行うことにより実現が可能である。

【００１４】
【数９】

【００１５】
【数１０】

$$f=A^{T}\cdot F\cdot A \quad \cdots（数１０）$$

【００１６】これらのことから、以下、説明を簡単にするため、主に１次元のｉＤＣＴを中心に説明を行う。

【００１７】図１，図２に一般的なｉＤＣＴ回路６の構成を示す。図１，図２は数５をもとに回路化した例である。直交変換においては入力されるデータ数と変換後のデータ数は同じである。そのため、ｉＤＣＴ回路の入出力条件としては、入力データレートと出力データレートは同じで、しかも間断なくデータを入力できることが望まれている。そのため、図１，図２の回路では数８の各行の演算を並列に行っている。

【００１８】図１では入力された信号１に対し、八つの積和演算器２－１～８で数２の積和演算を実行し、復号画像３－１～８（それぞれｆ（０）～ｆ（７）に対応）を得る。計算結果はブロックの計算終了後、速やかにそれぞれ記憶回路７－１～８に記憶され、積和演算器２－１～８は次のブロックのデータの処理を実行する。記憶回路７－１～８の信号８－１～８は選択器４で順に選択され出力画像信号５を得る。

【００１９】積和演算回路２は図２に示される構成をしている。入力信号１は制御回路１３の出力する係数信号１４と積算器１０で積算される。積算器１０の出力結果は記憶回路１２の内容１８と加算器１１において加算され、再び記憶回路１２に記憶される。ただし、各ブロックの第一のデータは加算器１１において加算処理されず、直接記憶回路１２に入力して初期化を行う。これらの積算・加算処理を入力データに同期して８回行うことにより数１の計算を行うことができる。係数信号１４は数８の行列の各行に対応する。すなわち、積和演算回路２－１では数８の１行目の係数が第一列目より順次用いられ、積和演算回路２－２においては２行目の係数が、以下同様に２－ｎの回路ではｎ行目の係数が用いられる。

【００２０】図３は上記で説明した積和演算回路２の動作タイミングチャートの形で示す。

【００２１】ＤＣＴの変換は数８を転置した行列を用いることにより上記で説明した逆変換と同じ回路により実行することができる。

【００２２】また、２次元のｉＤＣＴ（８×８）は図４

に示すように１次元ｉＤＣＴ回路６を縦続に接続することにより実現できる。２次元の入力信号は水平成分を優先してスキャンし、６４個の１次元信号５４として第一の逆変換回路２－１に入力される。逆変換回路２－１は６４個の入力信号を８個ずつ処理して行き、結果を６４信号分のメモリ５０に格納する。次にメモリ５０に格納された６４個の信号を２次元に並べ直し、さらに転置した後に再び１次元にスキャンして、第二の逆変換回路に入力する。このメモリ５０のデータ操作はメモリの読み書きのアドレスを変えることによって容易に実行することができる。

【００２３】即ち、入力は順にアドレス０，１，２，…８，９，…，６３と書き込み、出力は０，８，１６，３２，…，１，９，…，６３の順に読み出せばよい。第二の逆変換の結果復号された信号５５が得られる。入力を間断なく行うために、スイッチ５２，５０およびメモリ５１を付加し、２回の逆変換操作をパイプライン処理を行う。逆変換回路２－１の結果はスイッチ５２を介してメモリ５０に書き込まれている時に、逆変換回路２－２ではメモリ５１にある１操作前の結果をスイッチ５１を介して読み出す。一巡の操作が終了した時点でスイッチ５２，５３の接続先を変更することによりパイプライン処理を行うことができる。

【００２４】これらのｉＤＣＴあるいはＤＣＴ回路２の中の積和演算回路の数は、積和演算を入力データの８倍で動作させることにより、一つに減らすことも可能である。

【００２５】

【発明が解決しようとする課題】ＤＣＴおよびｉＤＣＴの計算は、八つの積和演算回路を用いることによりデータを間断なく処理する事ができるが、積算器の回路規模は非常に大きいため、ＤＣＴ回路あるいはｉＤＣＴ回路全体の回路規模が増大してしまう。一方、一つの積和演算回路のみで実行しようとすると従来の８倍の動作速度を必要とし、その実現が難しい。

【００２６】

【課題を解決するための手段】上記の課題を解決するには、積算に用いるデータを１ビットデータに分割し、積＊

＊算部分をシフト演算に置き換え、積和演算後に各ビットのデータを加算することによって実現できる。

【００２７】

【作用】上記の手段により従来と同じ動作速度で、しかも、回路規模も増加させることなしに変換あるいは逆変換処理を実行することができる。

【００２８】

【実施例】図５の実施例を用いて本発明の説明を行う。図５の実施例は図１と同じ動作を行うものである。図５中２５の部分が本発明の部分である。その詳細を図６および図７に示す。

【００２９】図５の演算のアルゴリズムを数１１～１８を用いて説明する。本発明では実数である行列の係数と入力信号の積算の方法が主な特徴となる。例えば、－１．３０６…の係数との積算を考えるとき、その絶対値の２進数表示は１．０１００１１ …となる。これを、

１．０００００

＋０．０１００００

＋０．０００１０

＋０．０００００１

の成分に分割した後に積算を実行する。各数値は最大一つの“１”のビットを有しているため、積算はシフト演算に置き換えることができる。入力信号をＦとしたとき、

Ｆ ＊ －１．０１００１１

を計算する代わりに、

－Ｆ ＊ １．０００００－ Ｆ ＊ ０．０１００００

－ Ｆ ＊ ０．００００１０－ Ｆ ＊ ０．０００００１

を計算する。

【００３０】数１１～数１３は数８の８×８の行列を７ビット精度の２進数表現した例である。

【００３１】

【数１１】

$$A = \frac{1}{2\sqrt{2}}(A_L\ A_R) \qquad \text{…（数１１）}$$

【００３２】

【数１２】

$$A_L = \begin{bmatrix} 1.000000 & 1.011000 & 1.010011 & 1.001011 \\ 1.000000 & 1.001011 & 0.100010 & -0.010001 \\ 1.000000 & 0.110010 & -0.100010 & -1.011000 \\ 1.000000 & 0.010001 & -1.010011 & -0.110010 \\ 1.000000 & -0.010001 & -1.010011 & 0.110010 \\ 1.000000 & -0.110010 & -0.100010 & 1.011000 \\ 1.000000 & -1.001011 & 0.100010 & 0.010001 \\ 1.000000 & -1.011000 & 1.010011 & -1.001011 \end{bmatrix} \qquad \text{…（数１２）}$$

【００３３】

【数１３】

7 8

$$A_R=\begin{bmatrix} 1.000000 & 0.110010 & 0.100010 & 0.010001 \\ -1.000000 & -1.011000 & -1.010011 & -0.110010 \\ -1.000000 & 0.010001 & 1.010011 & 1.001011 \\ 1.000000 & 1.001011 & -0.100010 & -1.011000 \\ 1.000000 & -1.001011 & -0.100010 & 1.011000 \\ -1.000000 & -0.010001 & 1.010011 & -1.001011 \\ -1.000000 & 1.011000 & -1.010011 & 0.110010 \\ 1.000000 & -0.110010 & 0.100010 & -0.010001 \end{bmatrix} \quad \text{...（数１３）}$$

【００３４】本発明では数１１～数１３の行列を、行列の各要素の中に有効なビット、即ち“１”であるビットが一つ以下になるように行列を分解している。数１４～数１８は数１２を分解した様子を示している。数１２の行列の各要素の絶対値を左のビット（Most Significant Bit：ＭＳＢ）より調べ、最初の“１”の部分のみを取り出した行列が数１５である。次に、数１２より数１５を引き同様にＭＳＢよりビットを調べ“１”の部分を取り出したものが数１６の行列である。数１６の行列は数１２の行列の各要素のＭＳＢより数えて２番目の“１”であるビットを示している。これらの操作を繰り返して＊ゆき、数１７，数１８を得る。数１８以降は行列の要素が全て０になるため行列の分解を中止する。この結果、数１２の行列は数１４のように四つの行列に分解される。

【００３５】

【数１４】

$$A_L = A_{L1} + A_{L2} + A_{L3} + A_{L4} \quad \text{...（数１４）}$$

【００３６】

【数１５】

$$A_{L1}=\begin{bmatrix} 1.000000 & 1.000000 & 1.000000 & 1.000000 \\ 1.000000 & 1.000000 & 0.100000 & -0.010000 \\ 1.000000 & 0.100000 & -0.100000 & -1.000000 \\ 1.000000 & 0.010000 & -1.000000 & -0.100000 \\ 1.000000 & -0.010000 & -1.000000 & 0.100000 \\ 1.000000 & -0.100000 & -0.100000 & 1.000000 \\ 1.000000 & -1.000000 & 0.100000 & 0.010000 \\ 1.000000 & -1.000000 & 1.000000 & -1.000000 \end{bmatrix} \quad \text{...（数１５）}$$

【００３７】

※ ※【数１６】

$$A_{L2}=\begin{bmatrix} 0.000000 & 0.010000 & 0.010000 & 0.001000 \\ 0.000000 & 0.001000 & 0.000010 & -0.000001 \\ 0.000000 & 0.010000 & -0.000010 & -0.010000 \\ 0.000000 & 0.000001 & -0.010000 & -0.010000 \\ 0.000000 & -0.000001 & -0.010000 & 0.010000 \\ 0.000000 & -0.010000 & -0.000010 & 0.010000 \\ 0.000000 & -0.001000 & 0.000010 & 0.000001 \\ 0.000000 & -0.010000 & 0.010000 & -0.001000 \end{bmatrix} \quad \text{...（数１６）}$$

【００３８】

★ ★【数１７】

$$A_{L3}=\begin{bmatrix} 0.000000 & 0.001000 & 0.000010 & 0.000010 \\ 0.000000 & 0.000010 & 0.000000 & -0.000000 \\ 0.000000 & 0.000010 & -0.000000 & -0.001000 \\ 0.000000 & 0.000000 & -0.000010 & -0.000010 \\ 0.000000 & -0.000000 & -0.000010 & 0.000010 \\ 0.000000 & -0.000010 & -0.000000 & 0.001000 \\ 0.000000 & -0.000010 & 0.000000 & 0.000000 \\ 0.000000 & -0.001000 & 0.000010 & -0.000010 \end{bmatrix} \quad \text{...（数１７）}$$

【００３９】

☆ ☆【数１８】

$$A_{L4}=\begin{bmatrix} 0.000000 & 0.000000 & 0.000001 & 0.000001 \\ 0.000000 & 0.000001 & 0.000000 & -0.000000 \\ 0.000000 & 0.000000 & -0.000000 & -0.000000 \\ 0.000000 & 0.000000 & -0.000001 & -0.000010 \\ 0.000000 & -0.000000 & -0.000001 & 0.000010 \\ 0.000000 & -0.000000 & -0.000000 & 0.000000 \\ 0.000000 & -0.000001 & 0.000000 & 0.000000 \\ 0.000000 & -0.000000 & 0.000001 & -0.000001 \end{bmatrix} \quad \text{...（数１８）}$$

【００４０】図５ではこれら数１５～数１８の各行列に 50 対して一つの１ビットのｉＤＣＴ回路２５－１～４が割

り当てられる。入力信号１は各ビット毎に分解された行列毎に演算されそれぞれの結果２６－１～４は加算器６３で加算され、最終結果を得る。

【００４１】図６は図５の１ビットのｉＤＣＴの計算回路である。信号の流れは図１のiDCT回路と同様であるが、積和演算の代わりにシフト加算回路２０－１～８が用いられている点、選択回路４の後段に定数倍回路２１の回路が挿入されている点が特徴である。

【００４２】図７は図６におけるシフト加算回路２０の詳細図である。入力された信号１はシフト回路３０において指定されたビット数ｎだけ右シフト、即ち２のｎ乗で除算される。シフト結果１５は加減算回路３１において過去の蓄積結果１８に対し加算、あるいは減算され、記憶回路１２に保持される。これらの処理を８回繰り返すことにより１ビットｉＤＣＴ回路２５の処理を行う。

【００４３】シフト回路３０におけるシフト数３３や、加減算器３１の加減算選択信号３４は制御回路３２により発生される。これらの内容およびタイミングの例を図８に示す。図８は数１５との演算を行う回路の例であ＊る。シフト数３３や加減算選択３４は八つのシフト加算回路においてもそれぞれ異なる。

【００４４】例えば、シフト加算回路２－１では図８に示すように、１番目の入力信号に対しては０、２番目に対しては０、以下順に、０，０，０，１，１，２が出力される。シフト数３３が０の時は入力信号を０ビット右シフト、即ち、数１５における１.０００００００を掛け、１の場合は１ビット右シフト、即ち０.１０００００を掛けた値を出力する。一方、加減算選択信号３４は図８の右のように加算(＋)か減算（－）かあるいは出力をゼロにするか（０）を選択する。

【００４５】これら、シフト加算した結果は記憶回路７－１～８に保持された後、選択回路４により順次選択される。選択された信号２３は、その選択した順に応じた出力係数が掛けられ出力される。各記憶回路７－１～８に対応する出力係数をａ０～ａ７としたときの演算行列を数１９に示す。

【００４６】

【数１９】

$$A=\begin{bmatrix} a0 & a0 & a0 & a0 & a0 & a0/2 & a0/2 & a0/4 \\ a1 & a1 & a1/2 & -a1/4 & -a1 & -a1 & -a1 & -a1/2 \\ a2 & a2/2 & -a2/2 & -a2 & -a2 & a2/4 & a2 & a2 \\ a3 & a3/4 & -a3 & -a3/2 & a3 & a3 & -a3/2 & -a3 \\ a4 & -a4/4 & -a4 & a4/2 & a4 & -a4 & -a4/2 & a4 \\ a5 & -a5/2 & -a5/2 & a5 & -a5 & -a5/4 & a5 & -a5 \\ a6 & -a6 & a6/2 & a6/4 & -a6 & a6 & -a6 & a6/2 \\ a7 & -a7 & a7 & -a7 & a7 & -a7/2 & a7/2 & -a7/4 \end{bmatrix} \quad \cdots (数１９)$$

【００４７】各出力係数ａ０～ａ７は全て同じ値でもよい。例えば、数１１の場合はａ０＝ａ１＝ａ２＝ａ３＝ａ４＝ａ５＝ａ６＝ａ７＝０.３５３６…(１／２／sqrt(2))の場合誤差が最も小さくなる。また、予め数１５～数１８の行列の代わりに数８を２進数表示し、ビット毎に分解した行列を用いることにより、ａ０＝ａ１＝…ａ７＝１として定数倍回路２１を削除する事も可能であり、本発明に含まれる。また、２次元ｉＤＣＴの場合は先の係数0.3536を２回掛けることになるため、これらを纏めて１回の積算にすることも本発明に包含される。即ち、１段目のiDCT回路の定数倍回路２１を削除し、２段目のｉＤＣＴ回路において２回分の定数倍を行う。先の例の場合には、０.３５３６…の２乗は０.１２５となるため定数倍回路は、３ビット右シフト回路に置き換えることができる。

【００４８】上記の行列の分解においては最大一つの“１”であるビットを持つ要素に分解し、計算した後に、加算あるいは減算を行ったが、加算と減算を組み合わせることによって計算回数を削減することが可能である。例えば、行列の係数が、２進数表示で０.１１１１１１　の時には、上記で説明した方法では６回のシフトと加算が必要であった。これを１.０００００００－０.０００００１と考えることにより２回のシフトと加減算で実行できる。このような最適化を行った場合も本発明に包含される。

【００４９】図７において係数が０である部分は計算を行わなくてもよい。例えば、一つの行の係数がすべて０であるような場合には図５においてその行に対するシフト加算回路２０を省略できる。また、一つの行の中で０でない係数が一つの場合には図７の加減算回路を省略することができる。このような最適化を行った場合も本発明に包含される。また、各１ビットｉＤＣＴ回路において最適化を行った結果、図５の２５－１～４のそれぞれの回路構成が異なってしまった場合も本発明に包含される。

【００５０】図９は本発明の第二の実施例である。図９は図５の実施例の１ビットｉＤＣＴ回路２５－２～４の部分を纏めてｉＤＣＴ回路６に置き換えた点が特徴である。また、計算順も１ビットｉＤＣＴ２５回路とｉＤＣＴ回路６が同時に計算するのではなく１ビットｉＤＣＴ回路２５が先に計算を行い、計算終了した後の任意のタイミングにｉＤＣＴ回路６が計算を開始することができる。１ビットｉＤＣＴ回路２５が先に近似計算を行い、その結果を信号線６７，スイッチ６５を介して出力し、後にｉＤＣＴ回路６において誤差信号を計算して先の近似値に加算し、正確な値をメモリ６４，スイッチ６５を介して出力する。

【００５１】１ビットｉＤＣＴの計算に用いる行列は図

１０に従い発生する。これは数２０の行列を用いて計算を行っていることになる。ｉＤＣＴ回路６では数８の行列の各要素から数２０の行列の各対応する要素の値を引いた値を要素とする行列を用いて演算を行う。これらの処理は特にソフトウェアなどを用いて順次にｉＤＣＴの計算を行う場合に有効である。まず、先に１ビットｉＤＣＴにより高速に近似計算を行い、次に正確な計算を行＊う。これにより、例えば、画像の復号化の処理では、処理開始直後にやや劣化のある画像が表示され、次第に劣化のない画像に置き換わって行くといった階層的な表示が可能である。

【００５２】

【数２０】

$$A=\begin{bmatrix} a0 & a0 & a0 & a0 & a0 & a0/2 & a0/2 & 0 \\ a1 & a1 & a1/2 & 0 & -a1 & -a1 & -a1 & -a1/2 \\ a2 & a2/2 & -a2/2 & -a2 & -a2 & 0 & a2 & a2 \\ a3 & 0 & -a3 & -a3/2 & a3 & a3 & -a3/2 & -a3 \\ a4 & 0 & -a4 & a4/2 & a4 & -a4 & -a4/2 & a4 \\ a5 & -a5/2 & -a5/2 & a5 & -a5 & 0 & a5 & -a5 \\ a6 & -a6 & a6/2 & 0 & -a6 & a6 & -a6 & a6/2 \\ a7 & -a7 & a7 & -a7 & a7 & -a7/2 & a7/2 & 0 \end{bmatrix} \quad \cdots (数２０)$$

【００５３】なお、図９において１ビットｉＤＣＴ回路２５は一つであるが、これを複数個用いて計算をし、その誤差をｉＤＣＴ回路６で計算することも本発明に包含される。例えば、図５の四つの１ビットｉＤＣＴ回路を順次計算し、加算した後に誤差成分をｉＤＣＴ回路６で計算し、それまでの計算結果に加えることにより、図５よりもさらに計算精度の高いｉＤＣＴを行うことが可能となる。

【００５４】図１０あるいは数２０で示される行列は先の図８あるいは数１９で示される行列を簡略化させたものである。３ビット以上シフトする部分を０に置き換え※ることによって計算量を削減している。

【００５５】図１１は図７の回路の変形例であり、図９の１ビットｉＤＣＴ部分に適用することにより効果的になる。図１１は図７の回路からシフト回路３０を削除したものである。これにより、例えば演算に使用する行列は数２１のようになり、計算誤差は数１９，数２０に比べやや大きくなるが、ソフトウェア処理では高速化が、ハードウェア処理では回路の簡略化を図ることができる。

【００５６】

【数２１】

$$A=\begin{bmatrix} a0 & a0 & a0 & a0 & a0 & a0 & a0 & 0 \\ a1 & a1 & a1 & 0 & -a1 & -a1 & -a1 & -a1 \\ a2 & a2 & -a2 & -a2 & -a2 & 0 & a2 & a2 \\ a3 & 0 & -a3 & -a3 & a3 & a3 & -a3 & -a3 \\ a4 & 0 & -a4 & a4 & a4 & -a4 & -a4 & a4 \\ a5 & -a5 & -a5 & a5 & -a5 & 0 & a5 & -a5 \\ a6 & -a6 & a6 & 0 & -a6 & a6 & -a6 & a6 \\ a7 & -a7 & a7 & -a7 & a7 & -a7 & a7 & 0 \end{bmatrix} \quad \cdots (数２１)$$

【００５７】図５，図９等の本発明の回路はｉＤＣＴの計算回路として説明を行ったが、先にも触れたように行列部分を変えることによりＤＣＴにも容易に適用可能である。即ち、例えば数２２に示す行列を計算することによりＤＣＴを計算することができる。先に示した実施例★および、変形例は全てＤＣＴの計算にも適用が可能である。

【００５８】

【数２２】

$$A=\begin{bmatrix} a0 & a0 & a0 & a0 & a0 & a0 & a0 & a0 \\ a1 & a1 & a1/2 & a1/4 & -a1/4 & -a1/2 & -a1 & -a1 \\ a2 & a2/2 & -a2/2 & -a2 & -a2 & -a2/2 & a2/2 & a2 \\ a3 & -a3/4 & -a3 & -a3/2 & a3/2 & a3 & a3/4 & -a3 \\ a4 & -a4 & -a4 & a4 & a4 & -a4 & -a4 & a4 \\ a5/2 & -a5 & a5/4 & a5 & -a5 & -a5/4 & a5 & -a5/2 \\ a6/2 & -a6 & ab6 & -a6/2 & -a6/2 & a6 & -a6 & a6/2 \\ a7/4 & -a7/2 & a7 & -a7 & a7 & -a7 & a7/2 & -a7/4 \end{bmatrix} \quad \cdots (数２２)$$

【００５９】図１２は図５の第三の実施例である。一つの１ビットｉＤＣＴ回路２５をその行列の係数を変えることにより、図５の１ビットｉＤＣＴ回路２５－１～４の計算を順次行って行く。それらの結果６６をメモリ６４に蓄積加算することによって図５と同じ結果を得ることができる。

【００６０】図１３は第四の実施例であり、図９の変形例である。図９においては１ビットｉＤＣＴ２５の計算結果をｉＤＣＴ６の結果に加算していたが、図１３の実施例ではｉＤＣＴ回路において正確なｉＤＣＴを行い、結果の加算を行わない。これにより図９より少ない計算量で、図９と同じ効果を得ることができる。

13

【００６１】以上説明したｉＤＣＴおよびＤＣＴの演算は各実施例そのものでなく、組み合わせた形でも実現できる。例えば、図５の実施例の１ビットｉＤＣＴ回路２５－１に数２１の行列を組み合わせることも可能である。

【００６２】図５において１ビットｉＤＣＴ回路には高速演算が適用可能である。１ビットｉＤＣＴ回路に高速演算を用いた場合も本発明に包含される。例えば、図１４は数２１の係数を用いる時の１ビットｉＤＣＴ回路の高速演算である。入力信号Ｆ（０）～Ｆ（７）は矢印で示されたデータどうし加算あるいは減算され丸印で示される中間結果、例えばＦ０２ｐ等、になる。これら中間結果もさらに加算あるいは減算され最終的にｆ（０）～ｆ（７）を得る。

【００６３】図１５は図１４の高速演算を行うソフトウェアの例である。Ｃ言語で書かれたプログラムの一部を示している。全ての変数は整数型で、名前は図１４の中間結果の名前に対応している。

【００６４】図１４の高速演算により、１次元のｉＤＣＴは加減算２４回と、定数倍の積算８回になる。また、２次元のｉＤＣＴを行っても加減算４８回と、３ビットのシフトを６４回行うだけで実行できる。

【００６５】図１６，図１７は本発明を用いた１次元／２次元のＤＣＴ回路およびｉＤＣＴ回路のブロック図である。２次元ＤＣＴ，ｉＤＣＴは数１０あるいは図４に示しているものと同じであり、２次元ＤＣＴの結果７２と中間結果である１次元のＤＣＴの結果７１をスイッチ７０で切り替えている。また、図１７のｉＤＣＴではスイッチ７３にて２次元ｉＤＣＴの入力７５と１次元ｉＤＣＴの入力７４を振り分け、１次元ｉＤＣＴの入力は２＊

14

＊段目のｉＤＣＴ回路６－２に入力している。なお、図１６のＤＣＴ回路６－３，４とｉＤＣＴ回路６－１，２は先に述べたように同じ回路で係数のみを変えることによって実現できる。図１６，図１７において問題になるのはＤＣＴ回路６－３，４，ｉＤＣＴ６－１，２の出力係数ａ０～ａ７の値である。１次元のＤＣＴ，ｉＤＣＴ回路ではａ０＝ａ１＝…ａ７＝０．３５３６…(＝１／２／sqrt（２）：sqrtは平方根を表す）である。しかし、行列の係数に実数を用いると直流成分のＦ（０）が実数値となり、計算精度によって誤差が生じやすくなり、復号化した画質にも誤差が見えやすくなる。そこで、１次元のＤＣＴの結果に予め２＊sqrt（２）（＝２．８２８４…）あるいはsqrt（２）（＝１．４１４２…）あるいは１／sqrt（２）（＝０．７０７１…）あるいは１／２／sqrt（２）(＝０．３５３５…）を乗じておき、逆にｉＤＣＴの時にこれらの値を除することによって、直流成分の計算が整数演算で実現でき、画質に最も影響を及ぼす直流成分の誤差をなくすることができる。以上をまとめると下のようになる。下の表は１次元ＤＣＴの結果を予め先の３通りの倍率にするときの各ＤＣＴ回路６－３，６－４およびｉＤＣＴ回路６－１，６－２における出力係数ａ０～ａ７の値(全て同じ値の場合)を示したものである。なお、表中ｓｑ２はsqrt（２）、即ち、１．４１４２ …を表す。Ａの項が数１，数２の定義通り、Ｂ，Ｃ，Ｄ，Ｅがそれぞれ２＊sqrt（２）倍，sqrt（２）倍，１／sqrt（２）倍，１／２／sqrt（２）倍に相当する。

【００６６】

【表１】

	ＤＣＴ		ｉＤＣＴ		１次元ＤＣＴの出力倍率
	６－３	６－４	６－１	６－２	
Ａ	1/2/sq2	1/2/sq2	1/2/sq2	1/2/sq2	１
Ｂ	１	１／８	１	１／８	２＊ｓｑ２
Ｃ	１／２	１／４	１／２	１／４	ｓｑ２
Ｄ	１／４	１／２	１／４	１／２	１／ｓｑ２
Ｅ	１／８	１	１／８	１	１／２／ｓｑ２

上記のＢ，Ｃ，Ｄ，Ｅが本発明に包含されることは明白である。またＢ，Ｃ，Ｄ，Ｅは積和演算回路を用いた１次元／２次元ＤＣＴ，ｉＤＣＴ回路にも適用可能である。

【００６７】図１６において１次元ＤＣＴ出力７１と２次元ＤＣＴ出力はスイッチ７０により切り替えられているが、これらを同時に出力することも本発明に包含される。図１７において入力切替スイッチ７３は入力直後に入っているが、スイッチ５２の直前に配置し、１段目のｉＤＣＴ回路６－１の結果と入力された信号とを切り替えてもよい。また、スイッチ７３は２段目のｉＤＣＴ回路６－２の直前に配置しても同様の効果が得られる。さらに、図１６と同様な構成をとり、１段目のｉＤＣＴの結果を１次元のｉＤＣＴの結果として出力してもよい。以上の変形例はすべて本発明に包含される。

【００６８】図１６あるいは図１７において信号を入力してから１次元ＤＣＴあるいはiDCTの結果が出るまでの時間と、２次元のＤＣＴ，ｉＤＣＴが出るまでの時間は異なる。１次元のＤＣＴあるいはｉＤＣＴの結果を遅延回路において遅延させて、２次元のＤＣＴあるいはｉＤＣＴの結果と同じタイミングに出力することができる。あるいは、１次元のＤＣＴあるいはｉＤＣＴの入力を遅延回路において遅延させることによっても同様の効果が得られる。これらの遅延回路は入力信号の入力順あるいは出力信号の出力順を変えるバッファと兼ねることも可能である。上記で述べた遅延回路を含む１次元／２次元

15

ＤＣＴあるいはｉＤＣＴ回路も本発明に包含される。

【００６９】

【発明の効果】直交変換の行列演算を１ビット毎に計算することにより従来と同じ動作速度で、しかも、回路規模も増加させることなしに変換あるいは逆変換処理を実行することができる。さらに、各ビットの計算をＭＳＢより順次行うことにより、ソフトウェア処理などの場合に概算値を先に高速で計算し、正確な値を順次計算することも可能になる。

【図面の簡単な説明】

【図１】従来の１次元ｉＤＣＴ回路のブロック図。

【図２】図１中の積和演算回路のブロック図。

【図３】図２のｉＤＣＴ回路の動作タイミングチャート。

【図４】２次元ｉＤＣＴの構成例のブロック図。

【図５】本発明による１次元ｉＤＣＴ回路のブロック図。

【図６】図５の１ビットｉＤＣＴ部分のブロック図。

【図７】図６のシフト加算回路のブロック図。

16

＊【図８】図６における制御信号の説明図。

【図９】本発明の第二の実施例のブロック図。

【図１０】図６における制御信号の第二の実施例の説明図。

【図１１】図７のシフト加算回路のブロック図。

【図１２】本発明の第三の実施例のブロック図。

【図１３】本発明の第四の実施例のブロック図。

【図１４】図１１の高速演算の信号フローチャート。

【図１５】図１３をソフトウェアで実現した例の説明図。

【図１６】１次元のＤＣＴと２次元のＤＣＴを一つの回路で実現するブロック図。

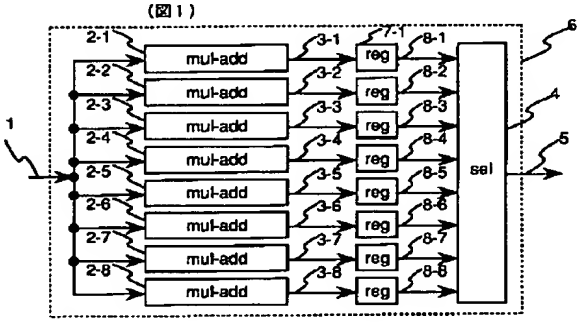
【図１７】１次元のｉＤＣＴと２次元のｉＤＣＴを一つの回路で実現するブロック図。

【符号の説明】

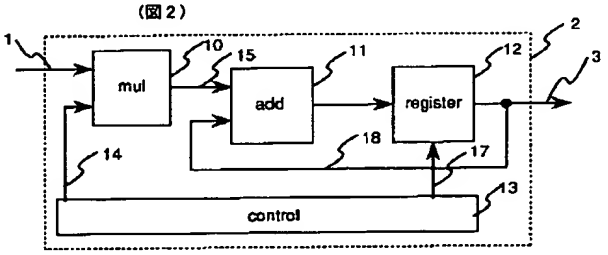
１…入力データ、２…積和演算回路、４…データ選択回路、５…出力データ、６…１次元ｉＤＣＴ回路、１０…積算器、１１…加算器、２５…１次元１ビットｉＤＣＴ回路、３０…シフト回路、３１…加減算回路。

＊

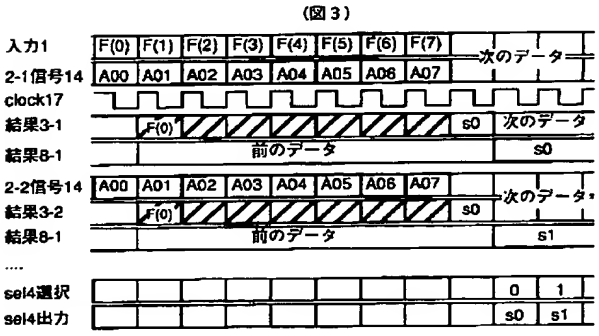
【図１】



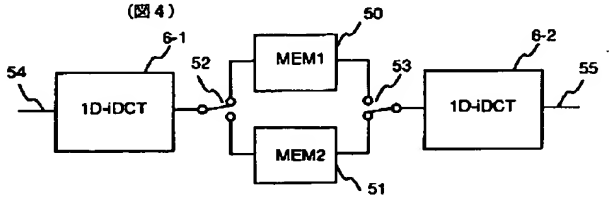
【図２】



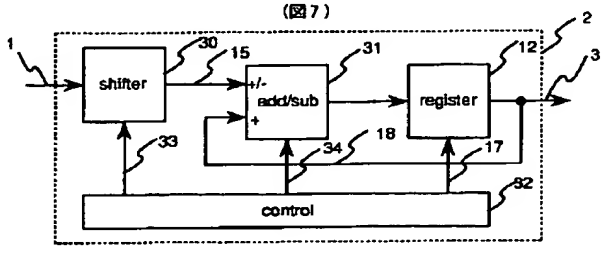
【図３】



【図４】

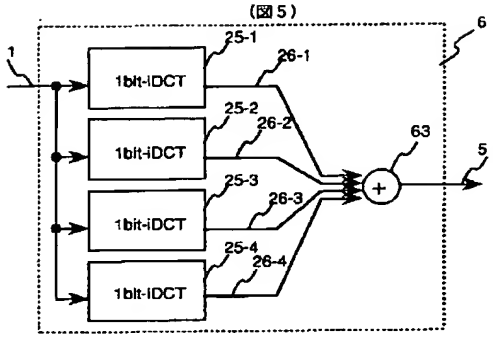


【図７】



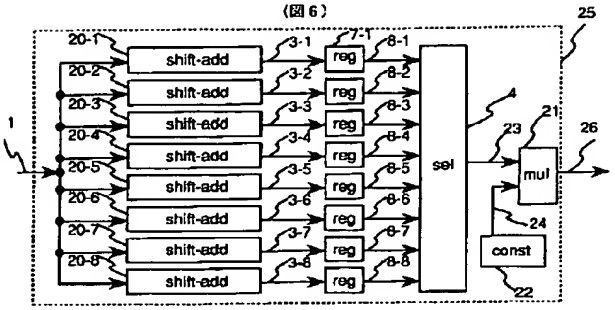
【図５】

(図５)



【図６】

(図６)



【図８】

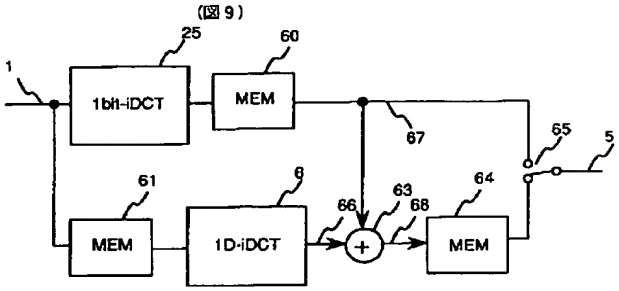
(図８)

入力1：	0	1	2	3	4	5	6	7
2-1の信号33：	0	0	0	0	0	1	1	2
2-2の信号33：	0	0	1	2	0	0	0	1
2-3の信号33：	0	1	1	0	0	2	0	0
2-4の信号33：	0	2	0	1	0	0	1	0
2-5の信号33：	0	2	0	1	0	0	1	0
2-6の信号33：	0	1	1	0	0	2	0	0
2-7の信号33：	0	0	1	2	0	0	0	1
2-8の信号33：	0	0	0	0	0	1	1	2

入力1：	0	1	2	3	4	5	6	7
2-1の信号34：	+	+	+	+	+	+	+	+
2-2の信号34：	+	+	+	−	−	−	−	−
2-3の信号34：	+	+	−	−	−	+	+	+
2-4の信号34：	+	+	−	−	+	+	−	−
2-5の信号34：	+	−	−	+	+	−	−	+
2-6の信号34：	+	−	−	+	−	−	+	−
2-7の信号34：	+	−	+	+	−	+	−	+
2-8の信号34：	+	−	+	−	+	−	+	−

【図９】

(図９)



【図１０】

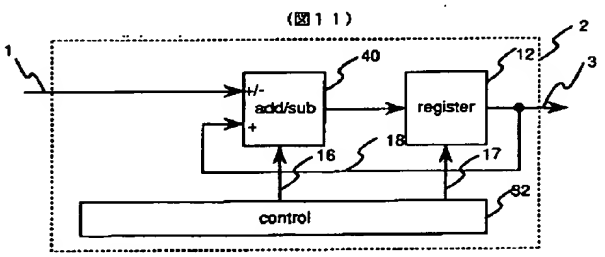
(図１０)

入力1：	0	1	2	3	4	5	6	7
2-1の信号33：	0	0	0	0	0	1	1	2
2-2の信号33：	0	0	1	2	0	0	0	1
2-3の信号33：	0	1	1	0	0	2	0	0
2-4の信号33：	0	2	0	1	0	0	1	0
2-5の信号33：	0	2	0	1	0	0	1	0
2-6の信号33：	0	1	1	0	0	2	0	0
2-7の信号33：	0	0	1	2	0	0	0	1
2-8の信号33：	0	0	0	0	0	1	1	2

入力1：	0	1	2	3	4	5	6	7
2-1の信号34：	+	+	+	+	+	+	+	0
2-2の信号34：	+	+	+	0	−	−	−	−
2-3の信号34：	+	+	−	−	−	0	+	+
2-4の信号34：	+	0	−	−	+	+	−	−
2-5の信号34：	+	0	−	+	+	−	−	+
2-6の信号34：	+	−	−	+	−	0	+	−
2-7の信号34：	+	−	+	0	−	+	−	+
2-8の信号34：	+	−	+	−	+	−	+	0

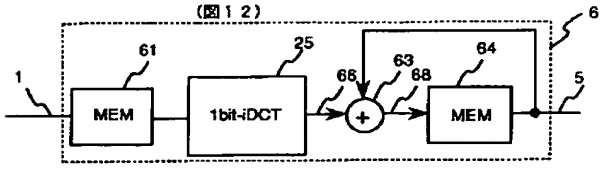
【図１１】

(図１１)



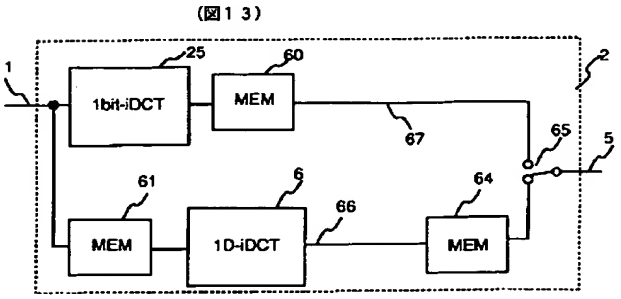
【図１２】

(図１２)



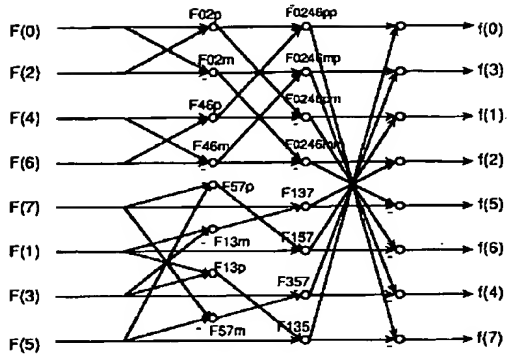
【図１３】

(図１３)



【図１４】

(図１４)



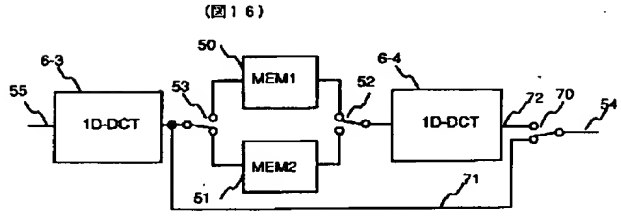
【図１５】

(図１５)

```
iDCT(int F[], int f[]){
  ....
  F02p = F[0] + F[2] ;
  F02m = F[0] - F[2] ;
  F46p = F[4] + F[6] ;
  F46m = F[4] - F[6] ;
  F13p = F[1] + F[3] ;
  F13m = F[1] - F[3] ;
  F57p = F[5] + F[7] ;
  F57m = F[5] - F[7] ;

  F0246pp = F02p + F46p ;
  F0246mp = F02m + F46m ;
  F0246pm = F02p - F46p ;
  F0246mm = F02m - F46m ;

  F135 = F13p + F[5] ;
  F157 = F[1] - F57p ;
  F137 = F13m + F[7] ;
  F357 =-F[3] + F57m ;

  f[0] = F0246pp + F135 ;
  f[7] = F0246pp - F135 ;
  f[1] = F0246pm + F157 ;
  f[6] = F0246pm - F157 ;
  f[2] = F0246mm + F137 ;
  f[5] = F0246mm - F137 ;
  f[3] = F0246mp + F357 ;
  f[4] = F0246mp - F357 ;
}
```

【図１６】

(図１６)



【図１７】

(図１７)